Leer datos de Internet

En el tema anterior aprendimos a leer flujos de archivos locales, lo mismo sucede con la lectura de archivos remotos, la única diferencia es que el flujo se abre en la red. Java cuenta con la clase `java.net.URL`, que te permite conectarte a un equipo remoto en Internet y acceder al recurso, siempre que no esté protegido del público.

Primero creas una instancia de la URL de tu recurso:

```
try {
    URL    xyz = new URL("http://www.xyx.com:80/training.html");
   …
}
catch (MalformedURLException e) {
   e.printStackTrace();
}
```

Se genera `MalformedURLException` si se especifica una URL incorrecta, por ejemplo, si se escribe **htp** en lugar de **http** o añade espacios sobrantes. Si se genera `MalformedURLException`, no indica que el equipo remoto tenga problemas, es decir, lo que debes comprobar es la ortografía de tu URL. La creación del objeto URL no establece una conexión con el equipo remoto; tendrás que abrir un flujo y leerlo. Sigue los pasos descritos a continuación:

- Crea una instancia de la clase `URL`.
- Crea una instancia de la clase `URLConnection` y abre una conexión por medio de la URL del paso anterior.
- Obtén una referencia a un flujo de entrada de este objeto mediante la invocación del método `URLConecction.getInputStream()`.
- Lee los datos desde el flujo. Usa un búffer para acelerar el proceso.

Durante el uso de flujos sobre la red, tendrás que controlar posibles excepciones de E/S como al leer archivos locales. El servidor al que intentas conectarte debe estar en funcionamiento y si usas protocolos basados en HTTP, un software especial, denominado servidor Web, debe escuchar al puerto especificado en el servidor. De esta forma predeterminada, todos los servidores Web escuchan solicitudes HTTP en el puerto 80 y las solicitudes HTTPS seguras se redirigen al puerto 443.

El programa del **listado 1** lee el contenido del archivo `index.html` de `google.com` e imprime el contenido en la consola del sistema. Para probar este programa, tu equipo debe estar conectado a Internet. Puede que el programa no funcione correctamente si la conexión debe atravesar un cortafuegos.

Listado 1. Leer el contenido index.html de google.com

```
import java.net.*;
import java.io.*;
public class WebSiteReader {
  public static void main(String args[]){
      String nextLine;
      URL url = null;
      URLConnection urlConn = null;
      InputStreamReader  inStream = null;
      BufferedReader buff = null;
      try{
        // index.html is a default URL's  file name
         url  = new URL("http://www.google.com" );
         urlConn = url.openConnection();
        inStream = new InputStreamReader(
                        urlConn.getInputStream(), "UTF8");
          buff  = new BufferedReader(inStream);

     // Read and print the lines from index.html
       while (true){
           nextLine =buff.readLine();
           if (nextLine !=null){
             System.out.println(nextLine);
           }
           else{
            break;
           }
        }
```

```
    } catch(MalformedURLException e){
      System.out.println("Please check the URL:" +
                                  e.toString() );
    } catch(IOException  e1){
     System.out.println("Can't read  from the Internet: "+
                                 e1.toString() );
    } finally{
            if (inStream != null){
             try{
                        inStream.close();
                        buff.close();
             } catch(IOException e1){
                         System.out.println("Can't close the streams: " + e1.getMessage());
             }
            }
    }
 }
}
```

El código del listado anterior crea una instancia de la clase URL, obtiene una referencia a una instancia de `URLConnection` para abrir una conexión al flujo y, por último, abre `InputStreamReader`, que se conecta a `BufferedReader`.

La clase WebSiteReader crea explícitamente el objeto URLConnection. Desde un punto de vista estricto, puedes obtener el mismo resultado con ayuda de la clase URL:

URL url = new URL(“http://www.google.com”);

InputStream in=url.getInputStream();

Buff=new BufferedReader(new InputStreamReader(in));

El motivo por el que usar la clase `URLConnection` es porque proporciona control adicional sobre el proceso de E/S. Por ejemplo, al invocar su método `setDoOutput(true)`, se especifica que este programa de Java también va a escribir en la URL remota. En el caso de conexiones HTTP, también establece el tipo de solicitud en POST. El método `setUseCaches(boolean usecaches)` de `URLConnection` también te permite especificar si el protocolo puede usar un objeto en caché o si siempre debe solicitar una copia nueva.

Por lo general, si tienes pensado crear programas de Java que funcionen con el protocolo HTTP, usa la clase `HttpURLConnection`, que admite funciones específicas de HTTP como el procesamiento de campos de encabezado, códigos de respuesta HTTP, establecimiento de propiedades de solicitud, etc.

Conectarse a través de servidores proxy http

Por motivos de seguridad, muchas empresas usan cortafuegos (visita http://es.wikipedia.org/wiki/Cortafuegos_(informática) para bloquear accesos no autorizados a sus redes internas. Como resultado, sus empleados no pueden acceder directamente a Internet (ni siquiera a determinados servidores internos) si no es a través de servidores proxy HTTP. Comprueba la configuración de tu navegador de Internet para ver si está protegido con un cortafuegos y saber el nombre del host y número de puerto del servidor proxy que utiliza. Normalmente, los navegadores web almacenan los parámetros de proxy en una ficha de opciones avanzadas en sus menús de Configuración o Preferencias.

Si tu navegador ha descargado una página que contiene un applet de Java, éste sabrá los parámetros de los servidores proxy y podrá realizar solicitudes a los servidores remotos a través del cortafuegos. Pero una aplicación estándar de Java debe especificar los parámetros `http.proxyHost` y `http.proxyPort` para poder atravesar el cortafuegos.

Por ejemplo, si el nombre de tu servidor proxy es `proxy.mycomany.com` y se ejecuta en el puerto 8080, debes añadir las siguientes líneas a tu aplicación para conectarla a Internet:

```
System.setProperty("http://proxyHost","http://proxy.mycompany.com" );
System.setProperty("http://proxyPort", 8080);
```

Si no quieres codificar estos valores, puedes pasarlos a tu programa desde la línea de comandos:

```
java –Dhttp.proxyHost=http://proxymycompany.com
        –Dhttp.proxyPort=8080 WebSiteReader
```

NOTA: –D<nombre>=<valor>: Para definir una propiedad del sistema

Otra opción consiste en usar la clase `java.net.proxy`. A continuación se muestra el código del mismo parámetro de servidor proxy (sustituye el nombre del servidor por una dirección IP):

```
Proxy myProxy=new Proxy(Proxy.Type.HTTP,
              new InetSocketAddress
                    (http://proxy.mycompany.com, 8080));
url=newURL("http://www.google.com/index.html");
urlConn=url.openConnection(myProxy);
```